Ingeniería del Software II.

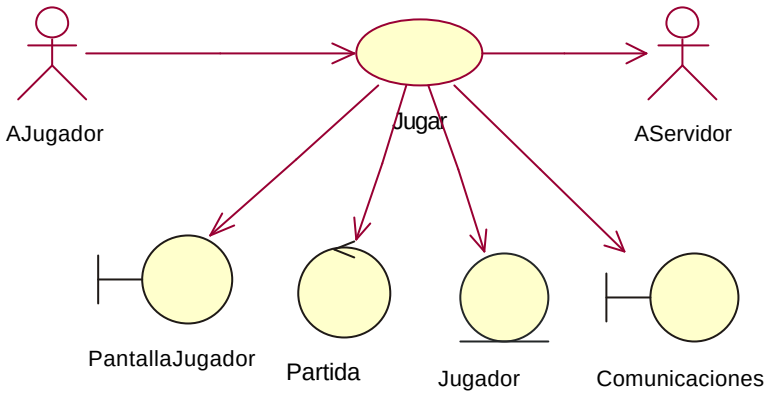
Desarrollo de proyectos OO.

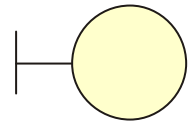
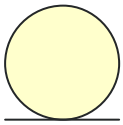
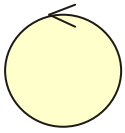
Como se recordará, en el Proceso Unificado de Desarrollo se distinguen cuatro fases principales:

1) Comienzo: es una fase secuencial en la que se preparan las iteraciones de las siguientes fases; es decir, se selecciona el orden de desarrollo de los casos de uso identificados.
2) Elaboración: en esta fase se realiza la planificación, el análisis y el diseño de la arquitectura del sistema. Se elabora cada caso de uso conforme al plan de iteraciones. Con cada caso de uso se detallan sus flujos de proceso, actores, diagramas de colaboración, posibles cambios de estado...
3) Construcción: se desarrolla y prueba el software de cada caso de uso. Puesto que el software vendrá totalmente diseñado de las fases anteriores, no habrá muchas decisiones de diseño.
4) Transición: el software está completo y se entrega a la comunidad de usuarios.

El desarrollo del software propiamente dicho se lleva a cabo en las fases 2ª y 3ª: es aquí donde se generan los diversos productos software (*artefactos*). Durante el análisis, que se encuentra aproximadamente en las fases de Comienzo y Elaboración, produciremos un *modelo de requisitos* y un *modelo de análisis*.

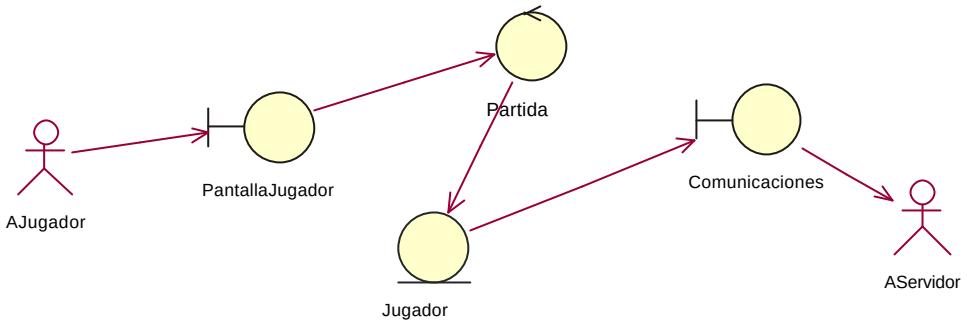
El modelo de requisitos se compone de los casos de uso, sus diagramas y documentación asociada. A partir de éstos, produciremos el modelo de análisis, que es un diagrama de "clases de análisis", para cuya descripción existe una notación específica:



 PantallaJugador	*Interfaz*: se utiliza para representar aquellos objetos que se van a comunicar con el entorno, que van a recibir estímulos y a enviar resultados a elementos situados fuera de los límites de nuestro sistema.
 Jugador	*Entidad:* denotan objetos que perviven. Suelen ser los objetos del dominio.
 Partida	*Control*: son objetos efímeros. Suele haber un objeto de control por cada caso de uso, aunque puede haber más de uno. Se les suelen asignar responsabilidades de control de transacciones, de secuencias, de aislamiento de objetos entidad, etc.

Centrados en un caso de uso, identificaremos en él las clases de tipo interfaz, entidad y control antes de pasar al siguiente caso de uso. Una misma clase puede participar en más de uno caso de uso, con lo que el proceso es iterativo.

Podemos utilizar las clase de análisis que intervienen en el caso de uso para representar con algo más de detalle el funcionamiento del caso de uso.



Cuando tenemos elaborados los modelos de requisitos y de análisis (tal vez no de todos los casos de uso), podemos pasar a diseñar el sistema.

Construiremos un modelo de diseño a partir del modelo de análisis y de la siguiente manera:

1) Teniendo en cuenta el entorno de implementación, lo que incluye el gestor de base de datos, comunicación con otros dispositivos, existencia y posible utilización de componentes o bibliotecas, tipos de gestión de errores, etc. El estudio del entorno de implementación puede hacerse paralelamente a la fase de análisis.
2) Construyendo una primera versión del modelo de análisis, que puede obtenerse construyendo una clase de diseño por cada clase de análisis.
3) Describiendo cómo interactúan los objetos de diseño en cada caso de uso. Para esto identificamos los escenarios y los eventos de cada escenario.

Inicialmente, habremos transformado cada clase de análisis en una clase de diseño, pero es probable que salgan más, obteniendo de esta manera subsistemas, paquetes, etc.. Refinaremos el modelo de diseño teniendo en cuenta el entorno de implementación. Éste afecta mediante factores como: lenguaje de programación, entorno de desarrollo, componentes y bibliotecas, disponibilidad de CASE, requisitos de rendimiento y memoria, características de la organización de desarrollo (estructura jerárquica, dispersión geográfica, cualificación...), etc. Estos factores producirán cambios en el diseño, pero todos los cambios deben estar justificados y documentados.

Pasaremos a la implementación cuando las interfaces de las clases comienzan a experimentar pocos cambios.

Durante todo el proceso habremos verificado y validado los artefactos producidos. Haremos lo mismo con el código, para lo que podemos aplicar técnicas específicas de prueba de programas:

1) Pruebas de unidad
2) Pruebas de integración
3) Pruebas de no regresión
4) Pruebas de aceptación (alfa/beta)
5) Inspecciones de código